# LUHbots RoboCup@Work 2017 Team Description Paper

Torben Carstensen[1], Andrej Dick[1], Jens Hübner[1], Alexander Wenz[1], Philipp Trusheim[1], Robin Kammel[1], Birgit Klein[1], Sven Falkenhain[1], and Jan Friederichs[2], Simon Aden[3]

[1] LUHbots, Leibniz Universität Hannover
Mechatronik Zentrum Hannover
Appelstraße 11, 30167 Hanover, Germany
`info@luhbots.de`
`http://www.luhbots.de`

[2] Hannover Centre for Mechatronics, Leibniz Universität Hannover
`friederichs@mzh.uni-hannover.de`

[3] Institute of Mechatronic Systems, Leibniz Universität Hannover
`daniel.kaczor@imes.uni-hannover.de`

**Abstract.** In this paper we provide a description of the LUHbots team. We describe the current state of the team as well as current plans and research goals towards the 2017 RoboCup@Work tournament in Leipzig. The software is based upon a ROS-architecture and the hardware uses a KUKA youBot as a basis. The focus of the team lies with failure tolerant approaches to mobile manipulation.

## 1 Introduction

The LUHbots team was founded in 2012 at the Institute of Mechatronic Systems Leibniz Universität Hannover, consists of bachelor and master students. Most of the founding team members have participated in the research inspired practical lecture RobotChallenge [10]. Nowadays the team is a part of the Hannover Centre for Mechatronics. The team consists of students from mechanical engineering, computer science, engineering, business administration and navigation and fieldrobotics. In 2012 the LUHbots team first competed in the RoboCup@Work challenge and was able to win the competition [9], in 2013 a second place was achieved [1]. In 2015 the LUHbots won both events, the German Open and the RoboCup in Hefei. In 2016 the team successfully completed the RoboCup@Work challenge in Leipzig (Germany), again achieving the first place.

## 2 Hardware

Our robot is based on the mobile robot KUKA youBot (see Fig. 1) [2]. The robot consists of a platform with four meccanum wheels [8] and a five degrees of freedom (DoF) manipulator. Additionally a gripper is attached at the end of the manipulator (see Fig. 1). The internal computer of the youBot has been replaced by an Intel Core i7 based system. In addition, the robot is equipped with an emergency stop system, allowing for keeping the platform and the manipulator in the actual pose when activated. The manipulator has been remounted to increase the manipulation area. The hardware itself does not offer failure tolerance, this is only achieved in combination with software.

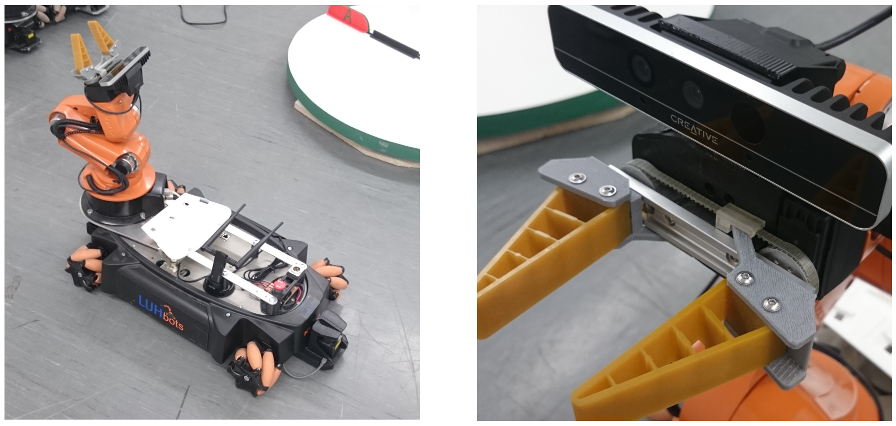

(a) LUHbots youBot (b) The youBot manipulator

Fig. 1: LUHbot 2017 - equipped with a new Gripper and the Intel RealSense F 200 camera.

### 2.1 Sensors

The youBot is equipped with two commercial laser range finders (Hokuyo URG-04LX-UG01) at the platform's front and back. A RGB-D camera (Intel RealSense F200) is mounted on the wrist of the manipulator (see Fig. 1a).

### 2.2 Gripper

One of the major hardware advances we performed is the development of a custom gripper. The original gripper has a low speed and stroke. As a result, it is not possible, to grasp all objects defined by the RoboCup@Work rule book without manually changing the gripper fingers. Besides the limited stroke, the low speed limit does not allow for an appropriate grasping of moving objects. As an advancement we included a force feedback into the gripper. Thanks to the integrated feedback within our custom made gripper, we are able to verify performed grasps. If a failure occurs during grasping, we are able to recover.

## 3 Approach

We take advantage of an open source software framework called Robot Operating System (ROS) [12]. We are using the Indigo release for 2017. In our opinion dependability is one of the most important aspects of mobile robots, therefore we are constantly improving our testing procedures.

### 3.1 Overview

Since our software architecture is based on ROS, different nodes are used (see Fig. 2). The yellow nodes are drivers they give access to the sensors. The youBot driver in red, can be accessed via the youBot OODL node. The camera data is first processed by the vision node and then filtered and clustered by the observer node, which is triggered by the state machine. The laser scanners are publishing to the navigation stack and the navigation watchdog. The watchdog filters the navigation commands. The task planner and the referee box connection communicate with the state machine. The laser scanner nodes are used unmodified. The ROS navigation stack is used but the global and local planners have been replaced. The youBot OODL driver is heavily modified. All other Nodes are developed entirely by the team.

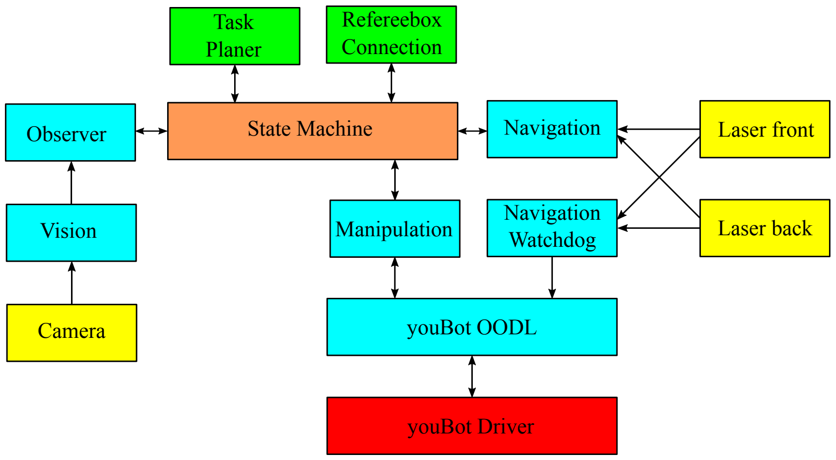
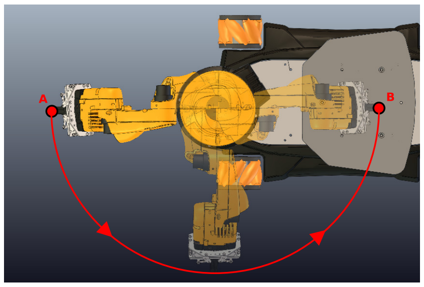


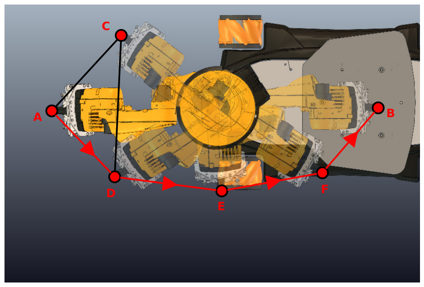
Fig. 2: Overview of the software architecture

### 3.2 Manipulation

During the last years we developed a new software system that can be seen as a software development kit (SDK) for manipulation tasks with the youBot. The aim was to facilitate the development of applications for the youBot by providing advanced functionality for the manipulator and the mobile platform combined with user friendly interfaces. Some of the features for the manipulator are: inverse kinematics, path planning, interpolated movement in joint- and task-space, gravity compensation and force fitting. Features for the mobile platform include incremental movement, collision avoidance and movement relative to the environment based on laser scans. The provided interfaces contain a documented API and a graphical interface for the manipulator. In the RoboCup we use this software e.g. to grab objects using inverse kinematics, to optimize trajectories and to create fast and smooth movements with the manipulator. Besides the usability the main improvements are the graph based planning approach (see Fig. 3) and the higher control frequency of the base and the manipulator. Planning on a graph which is based on known and, therefore, valid positions leads to a higher robustness. Using an A*-approach the best path is generated [6]. The higher frequency leads to better executed motion plans and an overall smooth and more accurate motion.



(a) Interpolation based planning



(b) Graph based path planning

Fig. 3: Graph based approach for the path planning, thanks to the proposed approach (b) a shorter motion is executed

## 3.3 Navigation

The navigation is based on the ROS navigation stack. we mainly improved the local and global planners. We extended the global planner to calculate the orientation for each pose of the global plan. This helps to reach the bottlenecks in the best position for a collision free and fast passing. Besides improving parts of the navigation stack we implemented a watchdog which operates based on the laser scanner data and is therefore much faster than a costmap-based local planner. The watchdog reduces velocities if an obstacle is too close or permits the execution of a movement command if a collision would be eminent.

### 3.4 Vision

We use the Intel RealSense F200 for object recognition, wich has one basic advantage in comparison to similar devices. Firstly it works at close range. We use the 2D-images of the infrared and RGB camera to segment the image to extract features and to classify the objects. We first seperate the objects from the infrared image using the canny algorithm [4]. Then we classify the objects using Hu-moments and a random forest classifier [13] [7]. Finally we determine the object's position and orientation using the 3D-points. Besides using the 2D-image we are developing a new Vision based on RGBD-point-clouds. The goal is that objects can be specified by standard CAD Files and an example texture. This would greatly increase the industrial usage, since it will be easy to include objects which the robot never had seen before. In order to get a robust vision system that can handle miss detections and which can memorize detected objects, we cluster all detections using a modified version of DBSCAN [5]. Each cluster is weighed, filtered and the positions are averaged. Then the clusters are classified as objects or as failures.

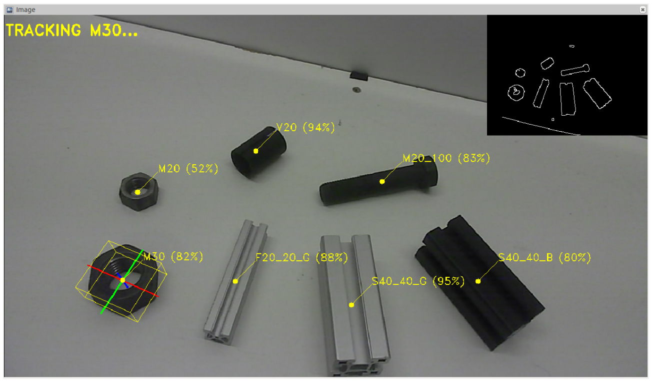


Fig. 4: Detected objects, classified and scored

### 3.5 Task planning

Our task planning is based on a graph based search. In each step all known service areas are used as possible navigation tasks. All objects on the back of the robot (there are up to three allowed) are used as possible placing tasks and the objects on the service area are used as grasping tasks. A greedy-based planning [11] is used up to a max depth and repeated until a complete plan is produced. The greedy algorithm is based on the cost function (see Eq. 1) taking the time to perform the task, the probability to fail and the expected output. For the navigation tasks the distances are precomputed based on the known map. The manipulation time costs are averaged based on the last respective manipulation action. When the state machine is not able to successfully recover a failure, the task is rescheduled and the probability to fail is increased.

$$Score_{n+1} = \frac{Value_n + Value_{Action} \cdot \prod Chance_i}{\sum Cost_i} \tag{1}$$

## 3.6 State machine

The state machine acts as an action client, which sets the goals in navigation and manipulation to accomplish the tasks and receives feedback in case of issues. Until last year our state machine was based on SMACH [3] which is a python library for building hierarchical state machines in ROS. For this year we aim to replace the SMACH state machine with our own c++ state machine. We want to preserve the modular capabilities of SMACH, but reduce the number of states and overall complexity that grew during the years. We also want to decrease errors during testing caused by run time errors due to python's run time compilation which cause a lot of wasted time in testing slots. We can then also merge the state machine with our manipulation state machine that is already coded in c++. The overall structure will persist and consists of the main components "task planning", "task execution," "navigation" and "manipulation" (see Fig. 5). The state machine is designed for recovery. If a failure is detected a direct recovery is applied. The current task will be adjusted, retried or postponed and performed again later, respectively.

## 3.7 Manipulation of dynamic objects

Our previous approach for gripping moving objects limited us. We could only grip two objects in one orientation, since the approach was based on calculating a point where to grasp the object and then performing a standard grasp. Because the speed was not used to alter the grasp motion the standard grasp resulted in problems with orientations. The new approach differs after the object recognition (see Fig. 6 - at time 1 ). The robot measures the speed and position of the object. It calculates the point and time where the object reaches the task place(see Fig. 6 - at time 2 ). The arm moves above the calculated point, waits for the object and accelerates until the arm is directly above the moving object with the same speed, then overlapping the down movement with the current speed until gripping the object (see Fig. 6 - at time 3 ). The advantage of this approach is that while the calculated position and speed are correct every orientation and much higher objects can be gripped.

## 4 Acknowledgements

We would like to thank a couple of institutes and persons supporting our work. The team is supported by the Institute of Mechatronic Systems, the Institute of Systems Engineering  Real Time Systems Group, the student affairs office of the

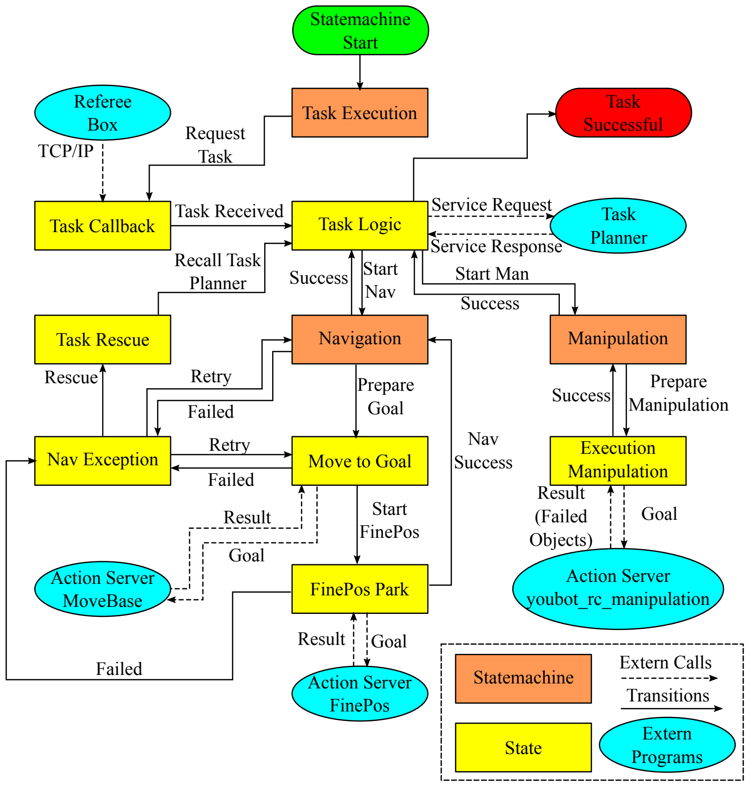


Fig. 5: State machine

faculty of mechanical engineering, the society for the promotion of geodesy and geoinformatics and the Hannover Centre for Mechatronics. The team is being supervised by Jan Friederichs, Johannes Gaa and Simon Aden.

## References

1. Alers, S., Claes, D., Fossel, J., Hennes, D., Tuyls, K., Weiss, G.: How to win robocup@work? the swarmlab@work approach revealed. In: RoboCup 2013: Robot World Cup XVII. pp. 147–158. Lecture Notes in Computer Science (2014)
2. Bischoff, R., Huggenberger, U., Prassler, E.: Kuka youbot - a mobile manipulator for research and education. In: ICRA (2011)
3. Bohren, J., Cousins, S.: The smach high-level executive [ros news]. IEEE Robotics & Automation Magazine 4(17), 18–20 (2010)
4. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8(6), 679–698 (Jun 1986)
5. Ester, M., Kriegel, H.P., S, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of 2nd International Conference on Knowledge Discovery and. pp. 226–231. AAAI Press (1996)

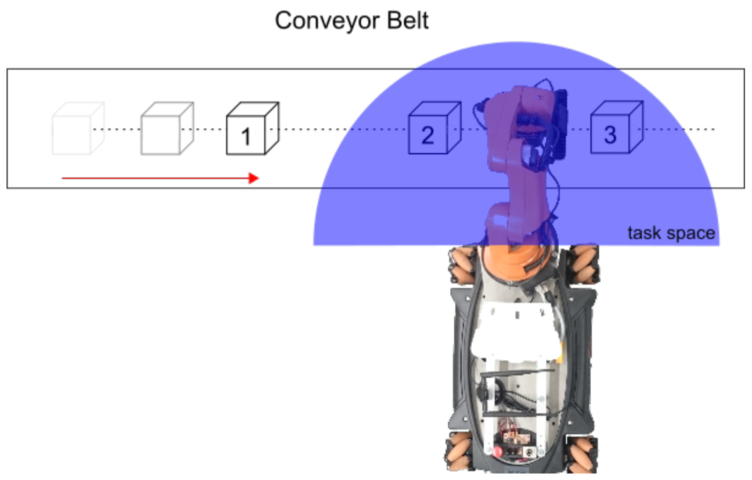


Fig. 6: Manipulation from the conveyor belt.

6. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. Systems Science and Cybernetics, IEEE Transactions on 4(2), 100 –107 (july 1968)
7. Hu, M.K.: Visual pattern recognition by moment invariants. Information Theory, IRE Transactions on 8(2), 179–187 (February 1962)
8. Ilon, B.: Directionally stable self propelled vehicle (Jul 17 1973), uS Patent 3,746,112
9. Leibold, S., Fregin, A., Kaczor, D., Kollmitz, M., El Menuawy, K., Popp, E., Kotlarski, J., Gaa, J., Munske, B.: Robocup@ work league winners 2012. In: RoboCup 2012: Robot soccer world cup XVI, pp. 65–76. Springer (2013)
10. Munske, B., Kotlarski, J., Ortmaier, T.: The robotchallenge - a research inspired practical lecture. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. pp. 1072–1077 (Oct 2012)
11. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1982)
12. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
13. Statistics, L.B., Breiman, L.: Random forests. In: Machine Learning. pp. 5–32 (2001)